Informe Trabajo Practico Progra2

[Gonzalo Tilca- León patricioleon92@gmail.com/gonzaloj16@hotmail.com Comisión 04 Patricio]

Introducción:

El objetivo de este informe es proporcionar una descripción detallada de la estructura de clases del sistema de gestión de servicios de una empresa de servicios. El sistema se encarga de administrar los diferentes tipos de servicios que ofrece la empresa, incluyendo servicios de electricidad, gasista, pintura y otros.

El diseño del sistema se basa en el paradigma de programación orientada a objetos, aprovechando los conceptos de herencia, encapsulación y polimorfismo para lograr una estructura modular y extensible. Cada tipo de servicio se representa mediante una clase específica, que hereda atributos y comportamientos de una clase base común.

El informe presenta una descripción detallada de las siguientes clases principales del sistema:

1. **Clase Cliente:** Representa a un cliente de la empresa de servicios. Contiene atributos como nombre, dirección y número de contacto, así como métodos para acceder y modificar esta información.
2. **Clase Empresa de servicios:** Representa a la empresa de servicios. Contiene atributos como nombre y dirección, así como una lista de clientes y servicios. Proporciona métodos para gestionar clientes, agregar servicios y generar informes.
3. **Clase Servicio:** Es la clase base para todos los tipos de servicios. Contiene atributos comunes como nombre, costo por hora y duración, así como métodos para calcular el costo total y generar una representación en cadena del servicio.
4. **Clase ServicioElectricidad:** Representa un servicio de electricidad. Hereda de la clase Servicio e incluye atributos adicionales como la potencia contratada y el tipo de conexión.
5. **Clase ServicioGasistaInstalacion:** Representa un servicio de instalación de gas. Hereda de la clase Servicio e incluye atributos adicionales como el tipo de instalación y la cantidad de artefactos.
6. **Clase ServicioGasistaRevision:** Representa un servicio de revisión de gas. Hereda de la clase Servicio e incluye atributos adicionales como la cantidad de artefactos y la fecha de la última revisión.

7. **Clase ServicioPintura:** Representa un servicio de pintura. Hereda de la clase Servicio e incluye atributos adicionales como el tipo de pintura y los colores.
8. **Clase ServicioPinturaEnAltura:** Representa un servicio de pintura en altura. Hereda de la clase **ServicioPintura** e incluye un atributo adicional para la altura de trabajo.

Cada clase se describe en detalle, incluyendo su ubicación en el paquete, atributos, métodos y relaciones con otras clases. Además, se proporciona una explicación de la herencia y cómo se utiliza en la estructura de clases para compartir y extender funcionalidades.

El informe tiene como objetivo proporcionar una visión general del diseño del sistema de gestión de servicios de la empresa, destacando la modularidad y la reutilización de código a través de la herencia. Esta estructura de clases permite una fácil expansión del sistema en el futuro, agregando nuevos tipos de servicios con mínimos cambios en el código existente.

Clase Cliente:

- Paquete: construcciones3

Atributos:

- **`dni`** (int): Representa el número de DNI del cliente.
- **`nombre`** (String): Almacena el nombre del cliente.
- **`numeroContacto`** (String): Guarda el número de contacto del cliente.

Métodos:

- **`Cliente(int dni, String nombre, String numeroContacto)`**: Constructor de la clase **`Cliente`** que recibe el DNI, nombre y número de contacto del cliente y los asigna a los respectivos atributos.
- **`getDni()`**: Método getter que devuelve el DNI del cliente.
- **`getNombre()`**: Método getter que devuelve el nombre del cliente.
- **`getNumeroContacto()`**: Método getter que devuelve el número de contacto del cliente.

La clase **`Cliente`** representa a un cliente de la empresa de servicios de construcción. Cada cliente tiene un identificador único (dni), un nombre y un numero de contacto. Esta clase se utiliza para almacenar la información de los clientes y realizar operaciones relacionadas con ellos, como obtener su información o realizar acciones específicas para cada cliente. Los conceptos utilizados son:

Encapsulación: Los atributos **`dni`**, **`nombre`** y **`numeroContacto`** de la clase **`Cliente`** están declarados como privados, lo que significa que solo pueden ser accedidos directamente desde dentro de la propia clase.

Métodos de acceso (getters): Los métodos **`getDni()`**, **`getNombre()`** y **`getNumeroContacto()`** son métodos de acceso que permiten obtener los valores de los atributos privados **`dni`**, **`nombre`** y **`numeroContacto`**, respectivamente. Estos métodos proporcionan una forma controlada de acceder a los atributos y obtener su valor.

IREP: Los atributos integer deben ser > 0 y los tipo String!=Null.

Clase EmpresaDeServicios:

Atributos:

- **`especialistas`**: Un mapa que almacena objetos de la clase **`Especialista`** utilizando el número de especialista como clave.
- **`clientes`**: Un mapa que almacena objetos de la clase **`Cliente`** utilizando el número de DNI como clave.
- **`servicios`**: Un mapa que almacena objetos de la clase **`Servicio`** utilizando el código de servicio como clave.
- **`serviciosFinalizados`**: Un mapa que almacena el precio total de los servicios finalizados utilizando el código de servicio como clave.
- **`codigoServicioActual`**: Un entero que representa el código del próximo servicio a ser registrado.
- **`facturacionTotalPorTipo`**: Un mapa que almacena la facturación total por tipo de servicio.

Métodos:

- **`registrarEspecialista(int numeroEspecialista, String nombre, String telefono, String`**

servicioEspecializado): Este método se utiliza para registrar un nuevo especialista en la empresa. Recibe como parámetros el número de especialista, nombre, teléfono y servicio especializado. Verifica si el número de especialista ya está registrado y si el servicio especializado es válido. Luego crea un objeto **Especialista** y lo agrega al mapa de especialistas.

- **registrarCliente(int dni, String nombre, String numeroContacto)**: Este método registra un nuevo cliente en la empresa. Recibe como parámetros el DNI, nombre y número de contacto del cliente. Verifica si el cliente ya está registrado y, de ser así, lanza una excepción. Luego crea un objeto **Cliente** y lo agrega al mapa de clientes.
- **solicitarServicioElectricidad(int dniCliente, int numeroEspecialista, String direccion, int horasTrabajo, double valorHora)**: Este método permite solicitar un servicio de electricidad. Recibe como parámetros el DNI del cliente, el número del especialista, la dirección, las horas de trabajo y el valor por hora. Verifica si el cliente y el especialista existen, si el especialista se especializa en electricidad y si los datos de electricidad son válidos. Luego crea un objeto **ServicioElectricidad** y lo agrega al mapa de servicios.
- **solicitarServicioPintura(int dniCliente, int numeroEspecialista, String direccion, double metrosCuadrados, double precioPorMetroCuadrado)**: Este método permite solicitar un servicio de pintura. Recibe como parámetros el DNI del cliente, el número del especialista, la dirección, los metros cuadrados y el precio por metro cuadrado. Verifica si el cliente y el especialista existen, si el especialista se especializa en pintura y si los datos de pintura son válidos. Luego crea un objeto **ServicioPintura** y lo agrega al mapa de servicios.
- **solicitarServicioPintura(int dniCliente, int numeroEspecialista, String direccion, double metrosCuadrados, double precioPorMetroCuadrado, int piso, double seguro, double alquilerAndamios)**: Este método es una sobrecarga del método anterior y permite

solicitar un servicio de pintura en altura. Recibe los mismos parámetros que el método anterior, además de los datos adicionales específicos para pintura en altura. Verifica si el cliente y el especialista existen, si el especialista se especializa en pintura en altura y si los datos son válidos. Luego crea un objeto **ServicioPinturaEnAltura** y lo agrega al mapa de servicios.

- **solicitarServicioGasistaRevision(int dniCliente, int numeroEspecialista, String direccion, int cantidadArtefactos, double precioPorArtefacto)**: Este método permite solicitar un servicio de revisión de artefactos de gas. Recibe como parámetros el DNI del cliente, el número del especialista, la dirección, la cantidad de artefactos y el precio por artefacto. Verifica si el cliente y el especialista existen, si el especialista se especializa en gasista y si los datos de revisión de gas son válidos. Luego crea un objeto **ServicioGasistaRevision** y lo agrega al mapa de servicios.
- **finalizarServicio(int codigoServicio)**: Este método se utiliza para finalizar un servicio dado su código de servicio. Calcula el costo del trabajo según el tipo de servicio, calcula el precio total a facturar al cliente (sumando el costo de materiales) y marca el servicio como finalizado. Guarda el precio total en el mapa de servicios finalizados y actualiza la facturación total por tipo de servicio.
- **cantidadDeServiciosRealizadosPorTipo()**: Este método cuenta la cantidad de servicios finalizados por tipo y devuelve un mapa con esta información.
- **facturacionTotalPorTipo()**: Este método devuelve la facturación total de un tipo de servicio dado su nombre.
- **facturacionTotal()**: Este método calcula la facturación total de la empresa sumando el precio total de todos los servicios finalizados.
- **cambiarResponsable(int codigoServicio, int nuevoResponsable)**: Este método se utiliza para cambiar el número de especialista responsable de un servicio dado su código de servicio. Verifica si el código de servicio y el nuevo número de especialista son válidos y si el tipo de servicio del nuevo especialista coincide con el tipo de servicio actual.

- **`listadoServiciosAtendidosPorEspecialista(int numeroEspecialista)`**: Este método devuelve un listado de los servicios atendidos por un especialista dado su número de especialista.

Clases y Objetos: Se definen varias clases, como **`EmpresaDeServicios`**, **`Especialista`**, **`Cliente`**, **`Servicio`**, **`ServicioElectricidad`**, **`ServicioPintura`**, **`ServicioPinturaEnAltura`**, **`ServicioGasistaRevision`** y **`ServicioGasistaInstalacion`**. Cada una de estas clases representa entidades del dominio del problema y tiene atributos y métodos asociados.

Mapas: Se utilizan diferentes objetos de tipo **`Map`** (HashMap en este caso) para almacenar y organizar la información. Los mapas se utilizan para almacenar los especialistas, clientes, servicios, servicios finalizados, facturación total por tipo, etc.

Excepciones: Se utilizan excepciones (RuntimeException) para manejar situaciones de error y lanzar mensajes de error personalizados en caso de que ocurra una excepción. Por ejemplo, se lanzan excepciones cuando se intenta registrar un especialista o cliente ya existente, cuando se solicita un tipo de servicio incorrecto, etc.

Herencia y Polimorfismo: Se utiliza la herencia y el polimorfismo en las clases de servicio. Las clases **`ServicioElectricidad`**, **`ServicioPintura`**, **`ServicioPinturaEnAltura`**, **`ServicioGasistaRevision`** y **`ServicioGasistaInstalacion`** heredan de la clase base **`Servicio`** y sobrescribe el método **`calcularImporteTotal()`** .

IREP: Los atributos integer deben ser > 0. Los maps con claves integer deben ser superior a 0 y las claves del tipo String deben ser distintos de null.

Clase Especialista:

- Paquete: construcciones3

Atributos:

- "numeroEspecialista": representa el número de identificación del especialista.
- "nombre": representa el nombre del especialista.
- "telefono": representa el número de teléfono del especialista.

- "servicioEspecializado": representa el tipo de servicio en el que el especialista está especializado.

Métodos:

- Especialista(int numeroEspecialista, String nombre, String telefono, String servicioEspecializado): constructor que recibe los datos del especialista y los asigna a los atributos correspondientes.
- getNombre(): String- devuelve el valor del atributo **nombre**.
- getNumeroEspecialista(): int- devuelve el valor del atributo **nombre**.
- getTelefono(): String - devuelve el valor del atributo **especialidad**.
- getServicioEspecializado(): String – devuelve el valor del atributo **servicioEspecializado**.

La clase **`Especialista`** es una representación de los especialistas que trabajan en la empresa de servicios. Cada especialista tiene un número de especialista, un nombre, un telefono de contacto y un servicio en un área específica. Se utiliza esta clase para almacenar la información de los especialistas y realizar operaciones relacionadas con ellos. En resumen, el código implementado representa un modelo de datos para un especialista en servicios, proporcionando los medios para almacenar y acceder a la información relacionada con dicho especialista. La clase "Especialista" encapsula los atributos necesarios y proporciona métodos de acceso para interactuar con los datos del especialista.

IREP: Los atributos integer deben ser > 0 y los tipo String!=Null.

Clase Servicio:

- Paquete: construcciones3

Atributos:

- "codigoServicio": integer que representa el código único del servicio.
- "dniCliente": integer que representa el número de identificación del cliente asociado al servicio.
- "numeroEspecialista": integer que representa el número de identificación del especialista asignado al servicio.

- "tipoServicio": string que representa el tipo de servicio que se está realizando.
- "direccion": string que representa la dirección donde se está llevando a cabo el servicio.
- "costoMaterial": double que representa el costo del material utilizado en el servicio.
- "tiempoTrabajado": integer que representa la cantidad de tiempo (en horas) dedicado al servicio.
- "finalizado": booleano que indica si el servicio ha sido finalizado o no.

Métodos:

- Servicio(intr. codigoServicio, int dniCliente, int numeroEspecialista, String tipoServicio, String direccion) El constructor de la clase se encarga de inicializar los atributos con los valores proporcionados, estableciendo los valores iniciales para " codigoServicio ", " dniCliente ", " tipoServicio ", " direccion " y " numeroEspecialista ".
- Los métodos "isFinalizado()", "setFinalizado()", "getCodigoServicio()", "getDniCliente()", "getNumeroEspecialista()", "getTipoServicio()", "getDireccion()", "getCostoMaterial()", "setCostoMaterial()", "getTiempoTrabajado()", "setTiempoTrabajado()", "setNumeroEspecialista()" son métodos de acceso y modificadores que permiten obtener y establecer los valores de los atributos de un servicio específico.
- El método "calcularImporteTotal(costomaterial)" devuelve un valor de tipo double y se espera que calcule el importe total del servicio derivando a sus métodos hijos el cálculo, pasando por parámetro el costo del material.

La clase **Servicio** representa un el conjunto de servicios de construcción ofrecido por la empresa. Tiene un nombre, un costo por hora y una duración en horas. Esta clase proporciona métodos para obtener información sobre el servicio, como el nombre, el costo por hora, la duración y el cálculo del costo total del servicio. La clase "Servicio" encapsula los atributos necesarios y proporciona métodos de acceso y modificadores para interactuar con los datos del servicio.

IREP: Los atributos integer y double deben ser > 0, los tipo String!=Null y el booleano "finalizado" se inicializa en false.

Clase ServicioElectricidad:

- Herencia: Esta clase hereda de la clase **`Servicio`**.

Atributos:

- "horasTrabajo": integer que representa la cantidad de horas trabajadas en el servicio de electricidad.
- "valorHora": integer que representa el valor monetario por hora de trabajo en el servicio de electricidad.

Métodos:

- ServicioElectricidad(int codigoServicio, int dniCliente, int numeroEspecialista, String tipoServicio,String direccion, int horasTrabajo, double valorHora): constructor que recibe el codigo del servicio, el dni del cliente, el número del especialista, el tipo de servicio, la dirección, las horas trabajadas que se van a utilizar y el costo de las mismas.
- getHorasTrabajo(): metodo que devuelve un int del valor del atributo **horasTrabajo**.
- getValorHora(): metodo que devuelve un int del valor del atributo **valorHora**.
- calcularCostoTotal(): sobrescribe el método de la clase padre para calcular el costo total del servicio de electricidad.

La clase **`ServicioElectricidad`** es una subclase de la clase **`Servicio`** y representa un servicio específico de electricidad ofrecido por la empresa de servicios. Tiene solo los atributos heredados de la clase padre los cuales van a tener el valor propio que le otorgue la clase hijo. Proporciona métodos para obtener información específica del servicio de electricidad, y también sobrescribe el método **`calcularCostoTotal()`** para tener en cuenta estos atributos adicionales en el cálculo del costo total del servicio.

IREP: Los atributos integer deben ser > 0.

Clase ServicioGasistaInstalacion:

- Herencia: Esta clase hereda de la clase **`Servicio`**.

Atributos:

- "cantidadArtefactos": integer que representa la cantidad de artefactos que se instalarán en el servicio de gas.
- "costoArtefacto": double que representa el costo unitario de cada artefacto.

Métodos:

- ServicioGasistaInstalacion(int codigoServicio, int dniCliente, int numeroEspecialista, String tipoServicio, String direccion, int cantidadArtefactos, double costoArtefacto): constructor que recibe el codigo del servicio, el costo por artefacto, la cantidad de los mismos, el codigo del servicio, el dni del cliente, el tipo del servicio a realizar y la direccion donde se realizara el trabajo.
- calcularCostoTotal(): sobrescribe el método de la clase padre para calcular el costo total del servicio de instalación de gas. Además del costo por hora y la duración para el cálculo.

La clase **`ServicioGasistaInstalacion`** es una subclase de la clase **`Servicio`** y representa un servicio específico de instalación de gas ofrecido por la empresa de servicios. Tiene solo los atributos heredados de la clase padre. Proporciona métodos para obtener información específica del servicio de instalación de gas, sobrescribiendo el método **`calcularCostoTotal()`** para tener en cuenta estos atributos heredados en el cálculo del costo total del servicio.

IREP: Los atributos integer y double deben ser > 0.

Clase ServicioGasistaRevision:

- Herencia: Esta clase hereda de la clase **`Servicio`**.

Atributo:

- "cantidadArtefactos": integer que representa la cantidad de artefactos que se revisarán en el servicio de gas.

- "costoArtefacto": integer que representa el costo unitario de cada artefacto.

Métodos:

- ServicioGasistaRevision(int codigoServicio, int dniCliente, int numeroEspecialista, String tipoServicio, String direccion, int cantidadArtefactos, double costoArtefacto): constructor que recibe el codigo del servicio, el dni del cliente, el telefono del especialista, el tipo de servicio a realizar, la direccion del cliente, la cantidad de artefactos y el costo de los mismos; posteriormente se asignan a los atributos correspondientes.
- calcularCostoTotal(): sobrescribe el método de la clase padre para calcular el costo total del servicio de revisión de gas. Además del costo por hora y la duración, también tiene en cuenta la fecha de la última revisión para aplicar un descuento si la revisión fue reciente.

La clase **`ServicioGasistaRevision`** es una subclase de la clase **`Servicio`** y representa el servicio específico de revisión de gas ofrecido por la empresa de servicios. Además de los atributos heredados de la clase padre, esta clase tiene atributos adicionales que giran en torno al precio y cantidad de artefactos por revision. Proporciona métodos para obtener información específica del servicio de revisión de gas como el método heredado **`calcularCostoTotal()`** para aplicar un descuento si la revisión fue reciente sobrescribiéndolo de la clase padre.

IREP: Los atributos integer deben ser > 0.

Clase ServicioPintura:

- Herencia: Esta clase hereda de la clase **`Servicio`**.

Atributos:

- "superficie":double que representa la superficie en metros cuadrados que se va a pintar.
- "costoMetroCuadrado":double que representa el costo por metro cuadrado de pintura.

Métodos:

- ServicioPintura(int codigoServicio, int dniCliente, int numeroEspecialista, String tipoServicio, String direccion,

double superficie, double costoMetroCuadrado): constructor que recibe el código del servicio, el dni del cliente, el telefono del especialista, el tipo de servicio a realizar, la direccion del cliente, la cantidad de metros cuadrados a trabajar y el costo de los mismos; posteriormente se asignan a los atributos correspondientes.

- calcularCostoTotal(): sobrescribe el método de la clase padre para calcular el costo total del servicio de pintura. Además del costo por hora y la duración, también tiene en cuenta el tipo de pintura utilizado para aplicar un costo adicional si es necesario.

La clase **`ServicioPintura`** es una subclase de la clase **`Servicio`** y representa el servicio específico de pintura ofrecido por la empresa de servicios. Además de los atributos heredados de la clase padre, esta clase tiene atributos adicionales que tienen la superficie en metros cuadrados a pintar y el costo del mismo. Proporciona métodos para obtener información específica del servicio de pintura, como la sobreescritura del método **`calcularCostoTotal()`** para tener en cuenta la superficie de pintura y aplicar un costo adicional si es necesario.

IREP: Los atributos double deben ser > 0.

Clase ServicioPinturaEnAltura:

- Herencia: Esta clase hereda de la clase **`ServicioPintura`**.

Atributos:

- "superficie": double que representa la superficie en metros cuadrados que se va a pintar.
- "costoMetroCuadrado": double que representa el costo por metro cuadrado de pintura.
- “piso”: integer que representa el piso en donde se realizara el trabajo.
- “costoSeguro”: double del costo del seguro por el trabajo de altura.
- “costoAlquielerAndamio”: double del costo del alquiler de los andamios.

Métodos:

- ServicioPinturaEnAltura(int codigoServicio, int dniCliente, int numeroEspecialista, String tipoServicio, String direccion, double superficie, double costoMetroCuadrado,int piso, double costoSeguro, double costoAlquilerAndamios): constructor que recibe el código del servicio, el dni del cliente, el telefono del especialista, el tipo de servicio a realizar, la direccion del cliente, la cantidad de metros cuadrados a trabajar y el costo de los mismos, además del piso en donde se va a trabajar, el costo del seguro y el costo de alquiler de los andamios; posteriormente se asignan a los atributos correspondientes.
- calcularCostoTotal(): sobrescribe el método de la clase **`ServicioPintura`** para calcular el costo total del servicio de pintura en altura. Además del costo por hora, la duración y el tipo de pintura, también tiene en cuenta la altura de trabajo para aplicar un costo adicional si es necesario.

La clase **`ServicioPinturaEnAltura`** es una subclase de la clase **`ServicioPintura`** y representa el servicio específico de pintura en altura ofrecido por la empresa de servicios. Además de los atributos heredados de la clase padre, esta clase tiene atributos adicionales que tiene son el piso en donde se efectúa el trabajo, el costo del seguro por trabajo en altura y el costo de los andamios alquilados. Proporciona métodos para obtener información específica del servicio de pintura en altura, como la sobrescritura del método **`calcularCostoTotal()`** para tener en cuenta la altura de trabajo y aplicar un costo adicional si es necesario.

IREP: Los atributos integer y double deben ser > 0.